A Conceptual Design of an Inattention Management Middleware with Adaptive Target Saliency

Max Nicosia and Per Ola Kristensson
Department of Engineering
University of Cambridge
Cambridge, United Kingdom

Abstract— **We present a conceptual design of an inattention management middleware with adaptive target saliency. The design objective is to provide mechanisms for managing operator inattention in multi-display multitasking applications. The conceptual design integrates ideas from situational awareness into its mechanisms to provide dynamic target saliency as a means to 1) guide operators through sub-tasks by drawing attention to high priority targets; and 2) guide operators on how to efficiently split their attention between tasks. We motivate the design by analysis of the results following a formative study with a prototype version of the conceptual design.**

TABLE OF CONTENTS

1. INTRODUCTION

Safety-critical system operators such as aircraft pilots, air-traffic controllers and drone operators are required to attend to, understand and process multiple data points of varying complexity in order to execute tasks effectively. Such situations present a risk to a performance as a result of operator inattention. Operator inattention can arise due to many factors, for example, information overload, cognitive overload, lack of training, stress, external distractions, and a general failure to factor in behavioral phenomena, such as change and inattentional blindness, into the design.

In this paper we propose a conceptual design of a middleware that can potentially mitigate such risks by actively managing operator inattention by dynamically changing the saliency of each relevant visualized data point to match its relevancy in relation to the operator's current task. Saliency can then be used as a means to steer the operator towards attending relevant information at the right time.

The conceptual design is based on abstracting an operator task into three task stages: 1) information detection (ID); 2) analysis and understanding (AU); and 3) task execution. These three stages are loosely based on a three-level model proposed by Endsley [1]. These task stage abstractions allow the middleware to be configured to trigger different saliency profiles for a specific data point dimension based on its relevance to the current task, the operators' performance, and its task stage. This design allows the middleware configuration to be decoupled from both the task and/or operator optimization profiles and the specific target application. This separation means that the middleware's configuration can be updated as the target application and its requirements evolve.

Our main contribution is the conceptual design of such attention-aware middleware, motivated in part by the results from a formative study.

2. RELATED WORK

The effects of interrupting users during task operation have been studied extensively in the human-computer interaction (HCI) field (e.g. [2], [3], [4]) and task specific fields such as air-traffic control research (e.g. [5], [6]). Commonly identified solutions involve avoiding interrupting the operator's current task [2] and using context-sensing and/or the contents of the message to infer a suitable moment to notify the operator [7], [4].

Design and development of attention-aware systems has been advocated as a promising way to provide general solutions to reducing interruptions and improving cognitive abilities of their operators [8], [9]. Previous efforts include a toolkit to manage attention by pushing notifications towards peripheral displays [8]. Prior work has also investigated other strategies, such as managing context switches, reducing interruptions, and tagging actively used objects [10], [11]. Roda and Nabeth [10] suggested using the tags assigned to specific resources as a way to guide how to manage interruptions and attention changes during a session with a learning system. They further suggest that the tagged resources could be used to prompt the learner to go back to the original resource after a certain time. In our design we use a similar approach, where we use saliency to guide user through the completion of their *Task* by guiding them towards *Targets* related to each *Task*. D'Mello et al. [11] showed that a learning system instrumented with eye-trackers to detect disengaged students was successful at redirecting their gaze back to the relevant material with the aid of auditory and visual cues. This supports our design approach.

Prior work has detected an operator's presence and position in relation to a set of displays and exposed it to proximity-aware applications[12], [13]. However, in more complex environments multi-sensing can be used to detect contextual information that allows to construct more optimal interruption strategies [7], [14]. Our design uses a similar multi-sensor approach. The system combines the output of eye-trackers to estimate the presence of the operator and then uses the gaze and head tracking information to accurately estimate the location of the operator's focus of attention.

Previous research has studied the effects of visualizing pertinent information in air-traffic control [6]. However, while visualizations improved performance of the main task, they had adverse effects on secondary tasks and did not generalize well [6]. Simple and subtle visualizations such as pulsating objects tended to perform better in general [6]. Recent work has investigated using eye-tracking data to increase the saliency of unseen changes in radar tasks [15]. The results were mixed, as several application-dependent factors, such as workload, task and the situation context, affected successful attention management [15]. This prior work has been one of our main motivators for including the adaptive saliency capabilities into our design.

Dostal et al [16] presented an inattention-aware multi-display system that used RGB cameras and computer vision to detect whether an operator was attending to a display or not, and used this information to derive subtle visualizations of unattended displays to reduce distraction while still allowing an operator to be aware of major display changes in their peripheral vision. [16]. Garrido et al [17] later distilled these research ideas into a toolkit with attention-aware graphical user interface (GUI) controls.

Nicosia and Kristensson [18] developed the Inattention Management Middleware (IMM), which provides several of the features previously discussed, such as fusing sensor data, detecting operator's focus of attention, detecting operators' target fixation and providing an interface for an application to specify callbacks on specific user events. The IMM provides an API that allows web-applications to detect whether specific data points have been looked at and allows the definition of application callbacks to be triggered on detection and data point intersection events. The API also provides a message passing mechanism to build inter-application information sharing and custom event logic, as well as functionality to push and cancel notifications among connected applications. This system is the starting point for the design described in this paper.

3. Problem Space

Air-traffic controllers have to perform several concurrent tasks of different levels of difficulty. Tasks include planning aircrafts' trajectories, monitoring their trajectories for correctness, reporting and recording various flight data statistics, and relaying and coordinating conflict resolutions with other air-traffic controllers and pilots [19].

To perform these tasks, air-traffic controllers are required to attend to various displays at the same time. These displays range from simple table-style information, such as weather forecasts, to complex displays that show multiple layers of data, such as radar data superimposed on geographical information and aircraft routes, trajectories, and meta-data for multiple aircrafts at the same time. We refer to each of these data point dimensions as an *information piece*. As multiple tasks are performed at the same time, information related to other tasks can become distractors. Part of the difficulty of an operator performing a task relates to an ability to filter out the right *information pieces* in order to make a correct decision.

In addition to the intrinsic complexity of said tasks, air-traffic controllers must adapt to various levels of difficulty of operation. Adverse weather effects, pilot skill levels, unforeseen crises, and general stress and tiredness of an operator can exacerbate task complexity. To mitigate these effects, air-traffic controllers undergo extensive training under a variety of different conditions and are instructed on best task prioritization procedures. However, even after extensive training, Seamster et al. [20] observed in their study that only the most experienced air-traffic controllers were able to prioritize more critical tasks reliably and maintain higher levels of situational awareness in demanding situations. Their study highlights that effective operation of such complex multitasks multi-display systems presents a real challenge for even very well trained personnel.

Figure 1. The two types of *Targets* that the hospital game used. The *Patient Target* (P): Top-left is the state, top-right is the health, bottom-left is the number of points to clear the Undiagnosed (U) or Extra Recovery (E) procedures, and bottom-right is the treatment plan. In the *Patient Target* above *UAE* means that the treatment path requires visiting the *Undiagnosed*, *Advanced Treatment* and *Extra Recovery* queues. The Doctor *Target* (D): Top-left is the healing and U and E points given per cycle, top-right is the health, bottom-left is the state, and bottom-right is the decay, or recovery rate.

To explain the motivations of our design and delimit our problem space, we have constructed a use-case that exemplifies all the challenges that a single operator of a real-world system can experience. We have deliberately removed the collaborative case from the use-case to limit the problem space to the single operator case. Defining our use-case allows us to create a requirements specification which we can then use to guide our design.

For our example use-case, we have chosen a game-like system that controls a simplified hospital where the four wings of said hospital are shown on each of the four displays. This choice of system metaphor was chosen as it allows for the relationship between the tasks to be simple to explain while also communicating the safety-critical nature of the tasks.

In this use-case, the operator's main task is to cure and discharge patients before their health runs out. The main *Objective* depends on the completion of two *Tasks*: 1) moving patients in a timely fashion through their various treatment destinations to eventually discharge them; and 2) allocating and resting doctors so they can continue to provide the necessary treatments to the patients in a timely fashion. Doctors provide treatment points and restore health at the expense of their own health. The patients lose health at a steady rate if they are not treated appropriately. If treatment is interrupted, patients continue to lose health. The two sub-tasks can only be performed effectively when the operator balances their attention and actions between the two tasks in a way that yields an acceptable turn-around rate.

Figure 2 shows each of the four displays with their various queues. Wing One (W1) contains the discharge queue, labeled *Home* and the three queues where patients arrive. Wing Two (W2) contains the Undiagnosed (U) queue, the Simple Treatment (S) queue and a doctor queue. Wing Three (W3) contains the doctor resting queue. Wing four (W4) contains the Advanced Treatment (A) queue, the Extra Recovery (E)

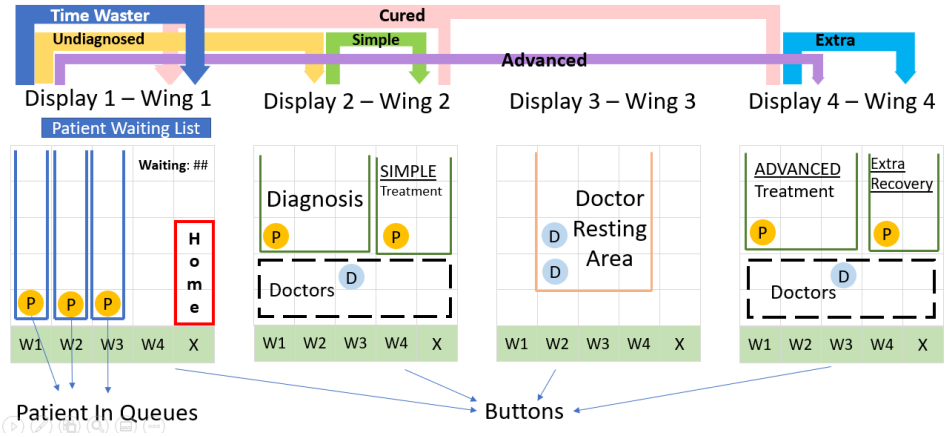

Figure 2. The four applications that constitute the hospital game example use-case.

queue and a doctor queue. The size of the queues represent the space limitations under which hospitals have to operate. Patients only stop losing health once they start undergoing one of the treatments. Idle patients and patients that are undergoing diagnosis will continue to lose health as they are yet to start treatment. Only after their treatment is finished do patients stop losing health.

Figure 2 shows the various routes the patients may need to take depending on their treatment requirements. Patients can arrive Diagnosed (D) or Undiagnosed (U). They may require Simple treatment (S), Advanced treatment (A) or no treatment at all. The last is an example of a hypochondriac patient. After their treatment, some patients may require an Extra Recovery (E) procedure before being discharged. In addition to their treatment plan, each patient also has a state, a health value and treatment requirement stats associated with it. Doctors have a health value, a healing value, which is directly related to their health value, a state, and a recovery or tiring rate, which will depend on their location or the number of patients they are treating. Figure 1 shows how our formative study displays the information for both doctors and patients. All these point dimensions are the *information pieces* that the operator needs to evaluate in order to execute the *Task* appropriately.

For operators to successfully cure patients before they lose all their health, they need to balance their attention between two sub-tasks: 1) moving patients through the treatment queues as required; and 2) allocating and resting doctors as required. Failure to move patients in a timely manner will lead to patients dying while waiting, or will prevent new patients from getting treatment due to lack of space. Similarly, poor doctor management will lead to doctors being depleted too fast, leading to patient death while they recover.

Optimal task performance requires the operator to contextualize the current operational situation and react differently. The operator needs to prioritize patients based on their treatment path, their current treatment stage, and their starting health. This is similar to how triaging works in hospitals. Similarly, doctors need to be rotated to ensure that their healing rates remain high enough to circulate patients at a rate that allows new patients to receive treatment in a timely manner.

To execute these tasks, the operator has to pay attention to the various values associated with each patient or doctor, and prioritize their next action accordingly. For example, normal operation will consist of watching a patient arrive and examining their health and treatment plan to decide what queue to allocate them to. Once their treatment is finished, the operator will proceed to move them to their next treatment stage so they can allocate their time to other tasks. Each sub-task is not complex in itself. However, operational complexity arise from both tasks having multiple concurrent instances. The operator then has to split their attention between competing tasks and their sub-tasks. The three well-known problems associated with such multitasks systems are present in our use-case as follows:

1. **Change Blindness [21]**: The operator fails to notice that a target's state has changed. For example, in our use-case a doctor that is low on health or a patient that is ready to be moved.
2. **Inattentional Blindness [22]**: The operator is too focused on one task such as allocating doctors and fails to notice that patients are about to expire.
3. **Information Overload [23]**: The operator fails to balance their attention or actions between the various sub-tasks. As a result of this, they lose track of what they are supposed to prioritize and they make mistakes. This may also occur as a result of stress when they perceive that there is too much going on.

Sustained acceptable performance will require the operator to develop a controlled strategy that will allow them to allocate their attention and time effectively. Such a strategy will consist of preempting the repercussions of their actions so they can make the best choices. The solution to this problem requires the operators to build and maintain a certain level of situational awareness during system operation in order to be effective. As such, we purposely built the idea of maintaining situational awareness into the design of our middleware. Our approach to this is discussed in the next section.

4. Middleware Design

The design of the proposed system integrates concepts of Situational Awareness (SA) and prioritization into its mechanisms to detect poor task performance and guide the operator towards a higher overall task performance through *Target* saliency manipulations. The system divided into four logical components that inter-operate with one another to provide the following functionality: 1) ease the operator's burden of noticing new information and finding information related to

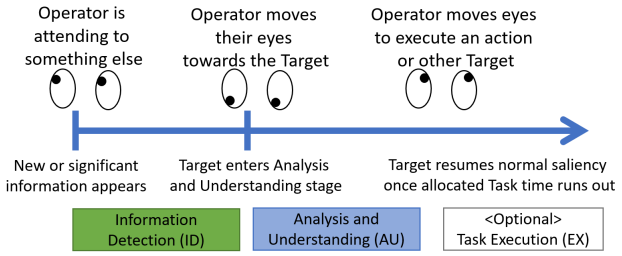

Figure 3. Diagram showing how the system tags *Target* based on state changes and operator actions.

or necessary for the current task; 2) identify poor task or sub-task performance; and 3) help the operator maintain an appropriate balance of time allocation between tasks.

Incorporating Situational Awareness into the Design

Figure 3 shows how the system tags *Targets* that require operator attention. When a point becomes a *Target*, that is, the system identifies it as relevant, the system tags its state as being in the *Information Detection (ID)* stage. In this stage, the system will increase the saliency of the *information pieces* that are relevant for identifying a *change*. Once the operator fixates on the *Target*, the system considers it as entering the *Analysis and Understanding (AU)* stage. In this stage, the system will manipulate the saliency of the *information pieces* that are relevant to executing the task. In the last optional stage, the system can use saliency to hint at specific actions that the operator can take. This will depend on system parameterization. Regardless of whether the last stage is used to manipulate the saliency of potential actions, the system can still collect data on an operator's response times for performance calculations.

The first two stages previously described are based on the first two levels of Endsley's three-level model [1]. Endsley's three levels are 1) Perception of elements in the environment; 2) Comprehension of the current situation; and 3) Projection of future status. The system's design does not explicitly have a stage for the Endsley's third level. Projection is indirectly managed by *Target* prioritization. We hypothesize that operators will make their own projection step as they iteratively perceive the saliency changes as a result of them approaching the *Anticipated* task performance. The idea of applying different saliency to the different pieces of information in a *Target* depending on whether it is in the *ID* or *AU* stage is based on concepts from Stanton et al.'s Distributed Situational Awareness [24]. We seek to ease the transactions between the agents, by easing the operator towards the information that they need for the task. The operator needs to know what information they need to find. However, by having the system prompting that information to the operator, he/she does not need to actively remember to look for it. We envisage that with correct parameterization, the changes in saliency can be made subtle enough to guide operators' attention without them realizing this in every case.

Target and Task Priority—As stated, the whole purpose of the system is to manipulate the operator's attention towards the most important *Target* related to the most important *Task* at a specific point in time. This is particularly relevant for operators of complex systems. Rantanen et al. [25] observed that air-traffic controllers prioritized tasks by categorical classification and not by quantifiable characteristics when under pressure. They also prioritize most recent tasks over older ones and simpler ones over the more complex ones.

When under strain, identifying the best *Target* can be difficult, especially if all the information is not easily accessible. For this purpose, we define *Target* importance in terms of the *Task* that will prevent critical failure and/or requires to be started first due to its time requirements. We express this concept of importance in terms of a *Priority* value. The *Task* with the highest *Priority* value will define what set of *Targets* the system tries to get the operator to attend to first. Similarly, the system will use the *Priority* value of the individual *Targets* within the *Task* to choose which ones to draw attention to first.

The design assumes that priorities are subject to change and that they will depend on operational circumstances. As such, priorities are continuously recalculated for both *Targets* and *Tasks*. In the case of *Task* priorities, the system uses them to decide the amount of time to allocate for its action, but it never lets higher priority actions starve lower priority ones. This concept follows directly from design principles for operating systems' resource schedulers [26]. By deliberately ensuring that *Tasks* are not starved, the system prevents *Tasks* from rising in priority artificially due to them being neglected. *Target* prioritization is taken care of by a specific sub-component, allowing for different prioritization mechanisms to be implemented depending on the domain.

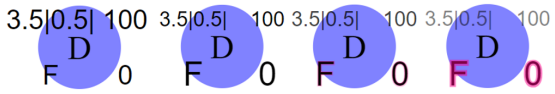

Figure 4. Different saliency levels used to draw attention to specific information within targets in the formative study. Top two values decrease in saliency level (0 to -20), while the bottom two values increase in saliency level (0 to 20).

Attracting Attention by Manipulating Saliency—The system uses saliency as a means to attract operator attention towards a particular *Target*. Saliency is represented as a continuous value from negative to positive within a predefined range. Negative saliency is used to make a *Target* or an information piece within it less noticeable. A zero value represents a neutral saliency level, while a positive value represents an increased saliency level. Figure 4 shows how saliency was manipulated in the formative study. The range was $[-20, 20]$. The top *information pieces* are at levels 0, -10, -11 and -20. The bottom ones are at saliency levels 0, 10, 11 and 20. Notice that between 0 and 10 only font size was affected, while between 11 and 20, the font turned transparent (in the negative case) or gained a pink outline (positive case).

The design also uses saliency to draw attention between different *information pieces* during the *ID* and *AU* stage changes of the *Target*. For example, an information piece can be highlighted during the *ID* stage and when the *Target* enters the *AU* stage, that information piece is reduced in saliency while other pieces are highlighted. This will prompt the operator to notice other relevant information while they are fixated on it.

Coordinated changes in saliency are achieved through *Saliency Profiles*. These are groups of saliency configurations that defines an ordered list of saliency changes that the system will apply in one iteration for a set of *Tasks* and their *Targets*. A *Saliency Profile* also defines the number of actions and time

limits per *Task* and *sub-Task*. This information is used by the system to ensure the operator does not fixate on just one *Task*.

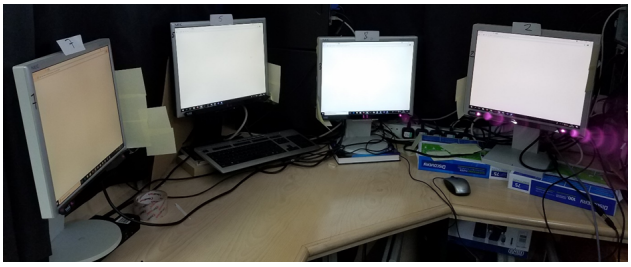

Figure 5. **The display setup used in the formative study.**

5. FORMATIVE STUDY

We carried out a formative study on a prototype version of the middleware to capture additional design insights. We designed the formative study as a within-subjects experiment with one independent variable with two levels: NOMITIGATIONS and MITIGATIONS. We recruited eight participants from the university. The average age was 32 years, ranging between 27 and 38 years ($sd = 4.3$). Six were male, the rest were female. Their eyesight was normal or corrected-to-normal, and they reported no motor or neural disorders. Participants were compensated with a £15 Amazon voucher for taking part.

All participants did two trials under each condition. *Trial 1* and *Trial 2*. Both trials had 45 simulated patients, arriving every other 8-second cycle. The trials only differed in patient order. The total number of doctors available for each trial was kept constant at 10. The difficulty of both trials was checked with a simulator we developed and both trials could be completed within fewer than 110 cycles ($\sim$15 minutes) using sub-optimal strategies. The trials were counter balanced using a latin square rule.

Mitigation Condition

Under the MITIGATIONS condition, the system triggered mitigation strategies that modified the saliency of the information within the *Targets* to draw attention to the *Target* and information that would allow the participant to infer the cause of poor performance.

Mitigations had one of two scopes: 1) *Local*, which measured a total or mean value of some quantity in one application. In this case one of the Hospital Wings; and 2) *Global*, which measured the total or mean of all application values. In this case the mean or total value of all the Wings in the Hospital that had that value in their Local Scope. The *Patient Task* had two *Global Mitigations* and the *Doctor Task* had four. In the *Local* scope, the *Patient Task* had four mitigations and the *Doctor Task* had six.

The difference in the number of mitigations between the *Doctor* and *Patient Tasks* was due to the fact that the mitigations were constructed to be triggered in response to specific performance measurements. Doctors had known states and stats, while patients' were unknown. As such, the mitigations for patients relate to their deaths in the *Local* and *Global* scopes, and discharge rates were in the *Global* scope only. Doctor mitigations relate to their health, rate of healing and states in both the *Local* and *Global* scopes.

Mitigations had different saliency levels for the affected *information pieces*. Some had negative saliency levels for some of the *information pieces* depending on their purpose. Mitigations had built in maximum periods to ensure they did not continue infinitely. Once their limit was reached, they entered a forced cool-down period that lowered their saliency level over each following cycle till they reached half their level. It was only after this cool-down process that the mitigation was allowed to trigger again if the performance was still determined to be below the *Anticipated* value.

Finally, all mitigations were added up and scaled before being applied to the *Targets*. The system applied scaling to ensure that saliency levels were constrained to the range $[-20, 20]$ across all applications. If any value was outside the range, all values were re-scaled back to the original range.

Apparatus and Materials

The experiment was carried out on four machines with an Intel i7 CPU and 8 GB of RAM running Windows 10. Each machine was fitted with a Tobii 4C eye-tracker and a display with a display area of 35 cm $\times$ 33 cm. The experiment was controlled and logged from a laptop with an Intel i7 CPU and 16 GB of RAM. All computers' clocks were synchronized using Windows Server 2019 running on a private network interconnected with a high-speed switch. Figure 5 shows the position of the displays. Their positions, heights and tilts were derived experimentally as to minimize interference between the eye-trackers. The IMM used in this formative was a more advanced version of our first prototype [18]. The one used in this experiment was capable of adapting saliency of *Target information pieces* depending on the active mitigations and the *Target's Task* stage. The web-application was written in *JS* using the *JS* API of the IMM in combination with the *d3.js* library [27] running on the Google Chrome Browser and two *C++* applications. The first application controlled trial instructions, and the second one logged all trial events.

Task

The experiment consisted of two interrelated tasks. The main task consisted of processing one resource (patients), subject to the availability of another resource (doctors). The secondary and competing task consisted of monitoring the doctors' health and resting them periodically as to not exhaust them, which would prevent them from treating more patients. The doctors' health directly affected the healing provided, and their effect was negated when doctors were exhausted. The appearance and information associated with doctors and patients can be seen in figure 1. The treatment plans (bottom-left) corresponded to the queues they needed to visit. Undiagnosed (*U*) patients needed to go to *W2*, while Diagnosed patients had to go directly to their treatment queue: Advanced (*A*) or Simple (*S*). Patients requiring Extra Recovery (*E*) needed to go to that queue before being sent home. Figure 2 shows the four wings (*W*) of the Hospital Game, the various queues in each, and the directions that patients could move based on their treatment plans. Doctors could be moved at any point between any of their queues.

Participants interacted with the system using a combination of gaze and a mouse held in their dominant hand. Participants selected individual visualized data points by gazing at the desired point while holding down the left mouse button until a green ring appeared around the point. Once the green ring appeared, participants could release the mouse button. Afterwards they could repeat this process to select more

points or move the current selection by selecting one of the Wing buttons at the bottom of the screen. Only data points of the same type could be selected at any one time, either doctors or patients. Movements were instantaneous. Incorrect movements were not prompted. Incorrectly moved patients were removed instantly, while doctors were moved back to Wing 3 after a short delay. Expired patients were highlighted in red for two seconds before being removed. Patients successfully sent *home* were highlighted in green before being removed.

Once a patient entered the right queue, its state changed from *X* (Not in Treatment) to *T* (Treatment). If the patient entered the wrong queue, or it its treatment was complete, its state was set to *X*. This ambiguity was by design to force participants to consider the context associated with the patient. A doctor's state was set to *W* (Working) while deployed, or *R* (Resting) if recovering health in *W3*. The extreme cases had their own states. Full health had the state set to *F* (100), and Exhausted was set to *X* (0).

The cycle tick length was set to 8 seconds. At the end of a cycle, all points in a Wing were updated and redrawn. Patients lost health at a rate of 5 points per cycle if they were not cured, receiving healing if they had not been healed to a 100 through the Simple or Advanced healing queues. Non-exhausted doctors provided 0.5 point of Diagnosis and Extra Recovery per cycle. They also provided 4 points of healing if their health was above 60, 3.5 points if between 20 and 60, 2 points if between 20 and 0, and 0 points if 0 (Exhausted). Doctors recovered 6.5 health points per cycle while resting and lost health while working. The health lost was given by calculating $\lfloor (\frac{1}{2} n_{\text{patients}}) + 1 \rfloor$.

The number of doctors available was fixed to 10, and their starting health values were spread as follows: three with 80, four with 50, and three with 20. This was done to ensure that participants would start to pay attention to doctors early in the experiment. The total number of patients was 45, and they were spread over 90 cycles. The rate of arrival was one every other cycle, with the last patient arriving in cycle 90. The treatment plans for the 45 patients were allocated as follows: 16 had *DAE*, 5 had *D*, 18 had *US* and 6 had *UA*. The patient's health was uniformly sampled from the range $[55, 70]$ and the values for *E* and *U* from the range $[5, 8]$.

Procedure

Participants were informed about the purpose of the experiment and how the eye-trackers operated. They were also told about the tracking limitations and that they needed to stay within the indicated distances to be correctly detected. Before explaining the task, participants were taken through the Tobii eye-tracking calibration procedure for each display as to ascertain if there were any problems with tracking. Only participants that presented no tracking problems were allowed to proceed.

The researcher carrying out the experiment explained the game in detail and ensured that the participant had understood the nuances of the various *information pieces* within each patient and doctor *Target* so they knew what to pay attention to. They were explicitly told that there were two competing tasks and that focusing exclusively on one of them would not allow them to be successful. This was reiterated to them to make sure that they understood that optimal operation required them to manage their resources wisely and not exhaust the doctors. They were also made aware that the starting doctors were not at 100% health. They were explicitly told to plan their recovery from the start of the Experiment. Participants were told that the game had a clock that ticked every eight seconds and that patients lost health every tick if not treated. They were also told that their best strategy was to prioritize patients with low health and, of those, the ones that were Undiagnosed. Participants were told that there was enough time if they prioritized patients accordingly. They were told that the lowest health of a patient was 55, and as such, they had about 1 minute and a half to get them to the right healing queue before their health would be fully depleted.

Participants were told that the experiment consisted of four trials and that they were free to take five minute breaks between them. Overall, the experiment did not take more than 1 hour 30 minutes including the training trial and calibration.

6. ANALYSIS

Patient Task Performance—We examined patient task performance under the two conditions with respect to the number of cycles that participants took to process all patients and the number of patients cured.

Table 1. **Formative Study Cycles to Process All Patients.**

Condition	Tr. No.	Mean	Std.	Min	Max
Mit.	1	107.1	5.0	102	117
Mit.	2	106.4	5.4	100	116
No Mit.	1	106.3	4.7	98	114
No Mit.	2	112.0	11.5	99	138

Table 1 shows descriptive statistics of the number of cycles that participants took to process 45 patients through the Hospital Game. With the exception of *Trial 2* under the NOMITIGATIONS condition, the mean, standard deviation, and maximum and minimum numbers of cycles taken under both conditions remained similar. This was expected, as both trials were constructed to be of the same difficulty. They both had the same patients with the same stats, only their order of appearance was changed. The rate of arrival was the same for both: every other 8-second cycle and the last patient arrived in cycle 90. The reason *Trial 2* under the NOMITIGATIONS condition being different is attributed solely to Participant 3, who took 138 cycles to complete that trial. An examination of the trace showed that this participant waited too long to send *cured* patients *home*, thus prolonging the experiment length. This result was outside our expectations as we estimated that most participants would finish within 110 cycles, given that patients expire after a period of time if not treated.

Table 2. **Expired Patients by Trial by Condition.**

Cond.	Tr. No.	Tot.	Mean	Std.	Min	Max
Mit.	1	67	8.4	6.4	2	18
Mit.	2	54	6.8	6.5	1	20
No Mit.	1	66	8.3	11.0	0	32
No Mit.	2	78	9.8	9.5	1	27

Table 2 shows descriptive statistics of the number of patients lost by the participants under each condition. In *Trial 1*, participants lost 67 $(sd = 6.4)$ patients under the MITIGATIONS

condition and 66 ($sd = 11.0$) under the NoMitigations condition. In *Trial 2*, participants lost 54 ($sd = 6.5$) patients under the Mitigations condition and 78 ($sd = 9.5$) under the NoMitigations condition. The number of lost patients is noticeable for *Trial 2*, but not for *Trial 1*. Participants lost 24 more patients in *Trial 2* under the NoMitigations condition. Under the NoMitigations condition losses are less consistent as suggested by the higher standard deviation.

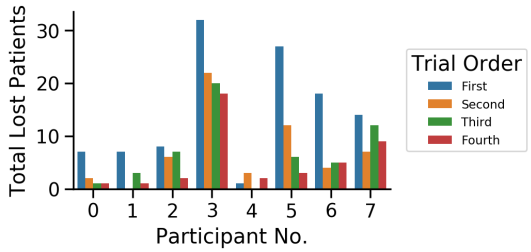

Figure 6. The total number of patients lost in each trial in the order they were performed.

Learning Effect—Figure 6 shows the number of patients lost by each participant in each trial in the order the trials were performed. A varying degree of learning can be seen in all participants. Participants 1 and 4 did not lose any patients in their second and third trials respectively. Learning was expected given that participants were only given 10–12 minutes to learn the task during the training trial. The learning effect was quite different across participants, highlighting that there may be broad differences amongst participants. Three participants commented that doing the Mitigations condition first helped them realize what information to focus on later on, indicating the possibility of an asymmetrical skill-transfer effect.

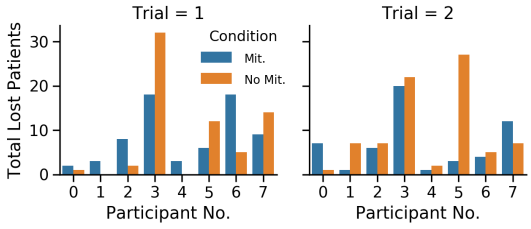

Figure 7. The total number of patients lost in each trial under each condition.

Figure 7 shows the number of patients lost in each trial under each condition. Participant 3 and 5 had the largest changes in *Patient Task* performance across conditions. Participant 6 appeared to have only struggled in their first trial and then continued to perform at a much better rate throughout (see Figure 6). Participant 7 performed inconsistently, as their performance switched between conditions. Participant 0, 1 and 4 exhibited consistently good performance throughout. Participant 2 had the same performance across three trials and improved in their last trial under the NoMitigations condition. About half the participants mentioned that they found the change of saliency between the *ID* and *AU* stage quite distracting at times. We discuss this further in section 7.

Patient Task Saliency Manipulation—We discuss here how the system reacted to the operator's performance and how this affected participants' performance. As mentioned in the *Mitigation Condition* Subsection in Section 5, there were several mitigations working at the same time. For the sake of space and brevity, we focus on the main mitigation affecting saliency in the *Patient Task* and the saliency changes in Wing 2. The mitigation in question triggered when patients were removed as a result of not being moved in a timely manner or being moved to the incorrect queue.

Figure 8 shows the running mean number of patients lost in Wings 1, 2 and 4 under both conditions. The blue line shows the Mitigations condition, while the orange shows the NoMitigations condition. The higher the value, the more patients that were lost. Patients expired mostly in Wing 2, as this was the location where participants had to diagnose patients before moving them into treatment. Focusing on the Mitigations condition (W2), there are peaks at around cycles 25, 50, 60 and 90 for *Trial 1*, and around cycles 15, 35, 45, 50, 70, 80 and 90 for *Trial 2*.

The mitigation worked by raising the saliency of information that could be useful to prevent the cause of low performance, while lowering the saliency of information not necessary at that stage. Figure 9 shows the mean saliency changes for both *ID* and *AU* for Wing 2 across all four *information pieces* of the *Patient Target*. The system's action consisted of increasing the saliency of *Health* and *State* while decreasing the saliency of *Decay* and *Diagnosis/Extra Cycles (D-E Cycles)*. The *Decay*, while informative, was designed to be mostly a distractor.

Figure 9 shows the mean saliency for each of the *information pieces* for the *Patient Target* in both trials under the NoMitigations condition. In *Trial 1*, there are peaks for *Health* at around cycles 25 and 60 for both *ID* and *AU* stages, and at around cycle 80 for the *AU* stage only. There is a large peak for the *ID* stage for the *State* at cycle 25, and it continues at varying degrees of a heightened level until cycle 70, where it decreases abruptly to start increasing again at cycle 80. Note that the *AU* stage does not follow the same pattern. This is because the mitigation was set up to draw attention to the change in state, that is, that the *Patient Target* was ready to be moved.

In *Trial 2*, the mean saliency of the *Health* in the *AU* stage does not increase as abruptly as in *Trial 1*. This is because the peaks in this trial are lower and more spread out. The system starts increasing the saliency more at cycle 45 where more peaks appear. The *ID* stage does not follow the same pattern in this case. This is probably due to other mitigations overloading the *ID* stage and reducing the effect of this mitigation, such as mitigations acting on *Doctor Targets* or patient *State*. This is expected as the operator only needs to know about the *Health* once the *Patient Target* is ready to move. The *State* matches the pattern previously described. The constant small peaks show the system maintaining a heightened level of saliency in the *ID* stage and lowering it for the *AU* stage.

Table 3. Total Mean Health Accumulated by Doctors in Wings 2 and 4 throughout the Formative Study for each Trial and Condition.

Cond.	Trial No.	Mean Total	Std.
Mit.	1	10903.3	2044.1
Mit	2	9863.0	2171.4
No Mit.	1	9842.6	2931.1
No Mit.	2	9604.9	2798.5

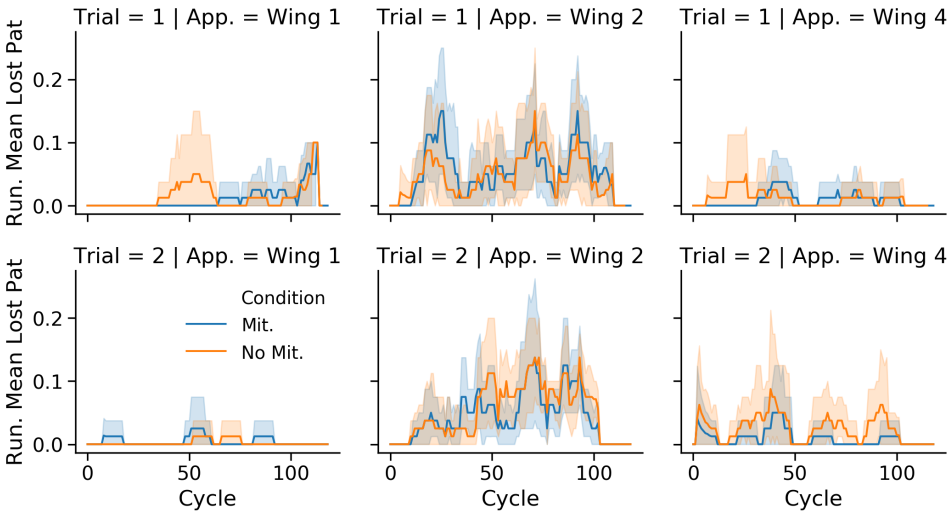

Figure 8. Running mean of the number of patients lost in each trial under each condition over the course of the experiment. Shading shows 95% confidence interval calculated from bootstrapping.

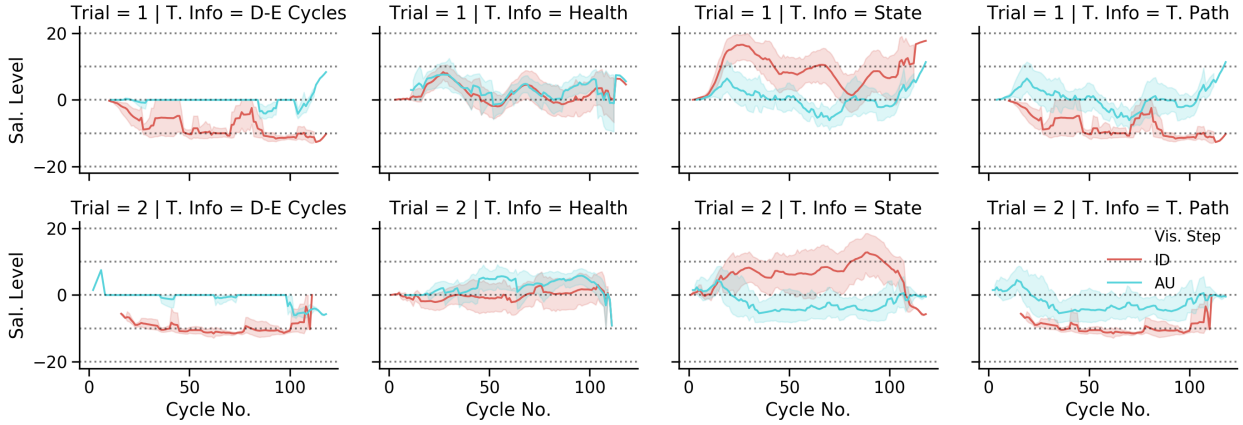

Figure 9. Mean saliency across trials for the MITIGATIONS condition for each information piece in the *Patient Target*. Shading shows 95% confidence interval calculated from bootstrapping.

Doctor Task Performance—Table 3 shows the total mean health accumulated by all doctors in Wings 2 and 4 across the formative study under each trial and condition. As discussed in the Subsection *Task* of Section 5, the higher the health of a doctor, the higher their healing power and the longer they can heal for. For this analysis, only Wings 2 and 4 are relevant. Including Wing 3 would boost the totals by adding the health values of doctors that were not scheduled to work. The MITIGATIONS condition increased the health values under both trials, with *Trial 1* having the largest increase. Participants were also more consistent as shown by the lower standard deviation across both trials under the MITIGATIONS condition.

Doctor Task Saliency Manipulation—Mitigations for doctors used a continuous running mean of the mean *Anticipated* health for all doctors in each Wing. If the value was below the tolerance, it would start triggering a mitigation to try to draw attention to the fact that the doctors would reduce their healing capacity as their health dropped. Figure 10 shows the *Anticipated* values in green, the measured values in blue and orange for the MITIGATIONS and NOMITIGATIONS conditions respectively. For both trials under the MITIGATIONS condition, the measured value is overall higher than

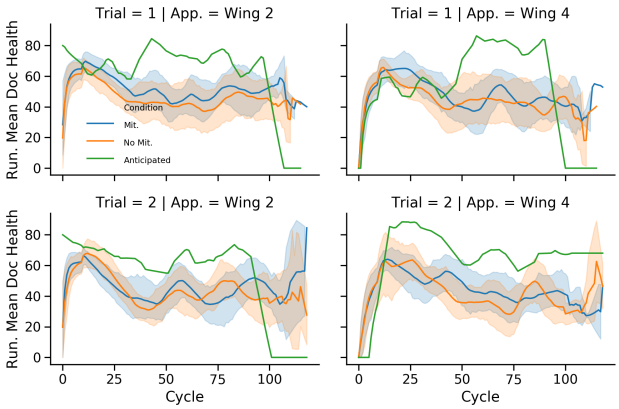

Figure 10. Running mean of mean doctor health under each condition and the running mean of the mean system's anticipated value over the course of the formative study for both trials in each Wing. Shading shows 95% confidence interval calculated from bootstrapping.

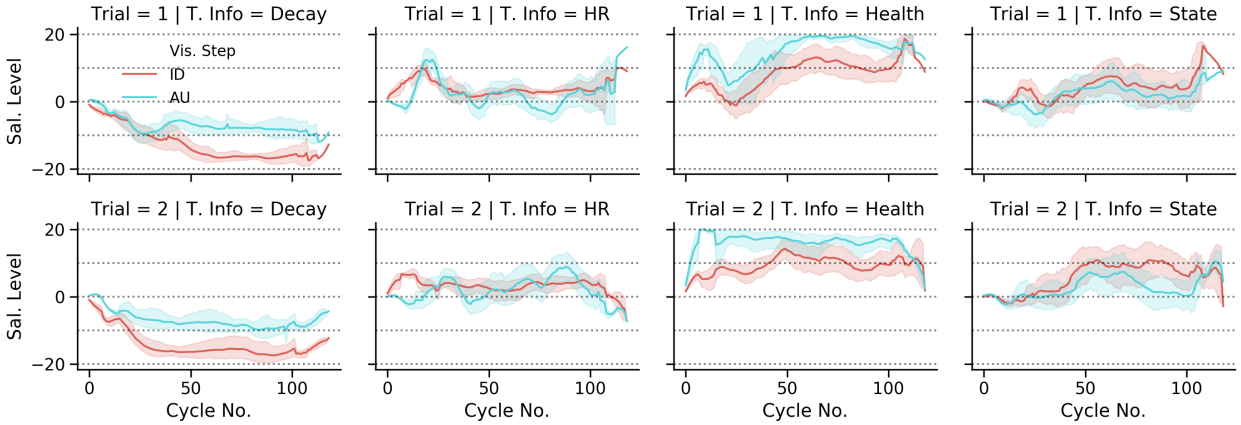

Figure 11. Mean saliency across trials for the MITIGATIONS condition for each information piece in the *Doctor Target*. Shading shows 95% confidence interval calculated from bootstrapping.

the one under the NOMITIGATIONS condition. This follows the results shown in Table 3. The *Anticipated* value was the value that the system used to trigger the mitigations. Beyond cycle 110, the measurements became less descriptive as participants started to finish the experiment at different points. Notice that while participants mirrored the *Anticipated* value, they were not always that close to it. Also, the *Anticipated* value decreased abruptly towards the end as it expected the actions for that Wing to finish. This did not affect the system's saliency as it applied cooldowns. We believe that the difference between the *Anticipated* and *Measured* values was due to how participants allocated their time between tasks. This is discussed in more detail in section 7.

As the *Doctor Task* in Wing 4 was the same as in Wing 2 we focus on Wing 2 for brevity and consistency. Figure 11 shows the mean saliency changes for both *ID* and *AU* for Wing 2 across all four *information pieces* of the *Doctor Target*. The system's action consisted of increasing the saliency of *Health* and *State* and sometimes *Healing Rate* while decreasing the saliency of the *Decay*. The *Decay*, while informative, was designed to be mostly a distractor. The saliency changes do not exactly match all the peaks. This is because, as mentioned in the subsection *Mitigation Condition* within section 5, there were several mitigations interacting with one another, so sometimes the effects of the measured changes in performance took several cycles to filter through.

Across both trials, the system started triggering the mitigation early as it expected participants to deploy patients to Wing 2 immediately. Many participants forgot to do this at the start. This resulted in the system increasing the saliency of the *Health* abruptly for both the *AU* and *ID* stage. Note that the *AU* stage is almost always higher than the *ID* stage. This is because changes in health were not significant at all points, only when they affected healing rates. The difference between the saliency patterns between the trials can be attributed to the system expecting participants to maintain a higher mean health. Notice that in *Trial 2* the saliency for both the *ID* and *AU* stages worked in lockstep with one another, while in *Trial 2* they were more independent. This follows from *ID* stage not being as important (within the mitigation) for *Health* so the system scaled it down.

The *State* saliency tended to follow saliency increases of *Health*. However, since its effect was set up to be less prominent than *Health*, it was normally scaled down due to *Health* being at a much higher saliency level. The *State* information was mostly relevant when it changed. As such, saliency was almost always higher for the *ID* stage than the *AU* stage.

7. OBSERVATIONS

Fixation Effects on Prioritization

One of the main problems identified from observing participants was that participants tended to address sub-tasks in batches. This was the result of poor prioritization due to task or application fixation. For example, participants would choose an application and try to complete all tasks within it, or choose a task and switch between all applications performing just that task or sub-task.

This strategy yielded mixed performance, as it did not allow for good prioritization of *Targets*, which led to poor time management between the *Tasks* and usually resulted in the participant attempting to catch-up. This behavior is clear in the differences observed between the *Anticipated* and *Measured* doctor health in Figure 10. The simulated operator rotated one doctor every 2–3 cycles, thus preventing a situation of having too many doctors with low health that would then needed to be shifted all at once. Similarly, the simulated operator always picked the best patient to move, instead of having to move all of them at the same time.

Saliency Overload

The prototype used in the formative study could sometimes end up overloading participants as it highlighted too many pieces of information at the same time. Participants identified two problems with the mechanism: 1) The pink outline used for highlighting the highest saliency targets was too distinct (see Figure 4); and 2) The saliency levels of the *ID* and *AU* stages could sometimes reach levels that were too far apart from one another, and the change between them was too distracting.

We anticipated some of these problems during implementation and built mechanisms to attenuate them. The scaling mechanism described in subsection *Mitigation Condition* in section 5 was specifically designed to reduce some of these effects. However, scaling did not solve the problem of separation between the *ID* and *AU* saliency levels, nor did

it stop too many *Targets* from switching between the *ID* and *AU* stages at the same time.

Removing Eye-trackers

In some specific domains, it may be possible to realize deployments without the eye-trackers if sufficient performance data is collected using a complete system with eye-trackers. In such cases, the system could detect the performance changes and trigger specific saliency profiles based on previously identified performance patterns. One caveat of this approach is that it will not be possible to make use different saliency profiles for the *ID* and *AU* stages. Also, detecting whether the operator has seen specific information will not be possible until they act on it.

8. Design Implications and Conclusions

From the prior analysis and observations we distill the following implications for design:

- Only a small number of *Targets* should have their saliency changed at any one time. The focused *Targets* should always relate to one *Task*, and their saliency should be intended to draw attention to their urgency, relevant information, and the order in which they should be addressed. To make this explicit, a subsequent *Target* have its saliency increased, at a lower level than the first *Target*. Once the first *Target* is addressed, the next target should be increased in saliency. This allows operators to identify the trajectory of *Targets* to address. A system should switch to highlighting the next *Task* and its *Targets* at a suitable time to ensure a suitable time allocation between *Tasks*.
- Ensure that the highest saliency level is only used when operators do not realize that they need to switch tasks.
- Ensure the saliency levels between *ID* and *AU* do not reach extremes by constraining them.

In summary, we have here described a conceptual design of a middleware for managing operator inattention with adaptive target saliency. The design is motivated from the literature, a representative use-case, and analysis and observations from a formative study. The formative study indicates that manipulating saliency can improve performance if care is taken to avoid overloading operators. Operators are in general better served when saliency is used on a subset of high priority targets in one task at a time. This ensures operators remain focused and split their time between tasks accordingly.

References

[1] M. R. Endsley, "Measurement of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 65–84, 1995. [Online]. Available: https://doi.org/10.1518/001872095779049499

[2] E. Cutrell, M. Czerwinski, and E. Horvitz, "Notification, disruption, and memory: Effects of messaging interruptions on memory and performance," in *Proceedings of Interact*, 2001, pp. 263–269.

[3] S. T. Iqbal and B. P. Bailey, "Effects of intelligent notification management on users and their tasks," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2008, pp. 93–102.

[4] A. Mehrotra, M. Musolesi, R. Hendley, and V. Pejovic, "Designing content-driven intelligent notification mechanisms for mobile applications," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 813–824.

[5] M. D. Wilson, S. Farrell, T. A. Visser, and S. Loft, "Remembering to execute deferred tasks in simulated air traffic control: The impact of interruptions." *Journal of experimental psychology: applied*, vol. 24, no. 3, p. 360, 2018.

[6] J.-P. Imbert, H. M. Hodgetts, R. Parise, F. Vachon, F. Dehais, and S. Tremblay, "Attentional costs and failures in air traffic control notifications," *Ergonomics*, vol. 57, no. 12, pp. 1817–1832, 2014.

[7] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mobile Networks and Applications*, vol. 7, no. 5, pp. 341–351, 2002.

[8] C. Roda and J. Thomas, "Attention aware systems: Theories, applications, and research agenda," *Computers in Human Behavior*, vol. 22, no. 4, pp. 557–587, 2006.

[9] B. P. Bailey and J. A. Konstan, "On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state," *Computers in human behavior*, vol. 22, no. 4, pp. 685–708, 2006.

[10] C. Roda and T. Nabeth, "Supporting attention in learning environments: Attention support services, and information management," *Creating New Learning Experiences on a Global Scale*, pp. 277–291, 2007.

[11] S. D'Mello, A. Olney, C. Williams, and P. Hays, "Gaze tutor: A gaze-reactive intelligent tutoring system," *International Journal of Human-Computer Studies*, vol. 70, no. 5, pp. 377–398, 2012.

[12] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg, "The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 2011, pp. 315–326.

[13] J. Dostal, U. Hinrichs, P. O. Kristensson, and A. Quigley, "Spidereyes: designing attention-and proximity-aware collaborative interfaces for wall-sized displays," in *Proceedings of the 19th International Conference on Intelligent User Interfaces*, 2014, pp. 143–152.

[14] H. Lopez-Tovar, A. Charalambous, and J. Dowell, "Managing smartphone interruptions through adaptive modes and modulation of notifications," in *Proceedings of the 20th International Conference on Intelligent User Interfaces*, ser. IUI '15. New York, NY, USA: ACM, 2015, pp. 296–299. [Online]. Available: http://doi.acm.org/10.1145/2678025.2701390

[15] B. R. Vallières, H. M. Hodgetts, F. Vachon, and S. Tremblay, "Supporting dynamic change detection: using the right tool for the task," *Cognitive research: principles and implications*, vol. 1, no. 1, p. 32, 2016.

[16] J. Dostal, P. O. Kristensson, and A. Quigley, "Subtle gaze-dependent techniques for visualising display changes in multi-display environments," in *Proceedings of the International Conference on Intelligent User Interfaces*, 2013, pp. 137–148.

[17] J. E. Garrido, V. M. Penichet, M. D. Lozano, A. Quigley, and P. O. Kristensson, "Awtoolkit: attention-aware user interface widgets," in *Proceedings of the 2014 Interna-*

tional Working Conference on Advanced Visual Interfaces, 2014, pp. 9–16.

[18] L. Nicosia and P. Kristensson, "Inattention-management middleware for human-in-the-loop multi-display applications," *In Proceedings of the IEEE Workshop on Human-Centered Computational Sensing (HCCS 2018)*, 2018.

[19] J. M. Histon, R. J. Hansman, B. Gottlieb, H. Kleinwaks, S. Yenson, D. Delahaye, and S. Puechmorel, "Structural considerations and cognitive complexity in air traffic control," in *Proceedings. The 21st Digital Avionics Systems Conference*, vol. 1. IEEE, 2002, pp. 1C2–1C2.

[20] T. L. Seamster, R. E. Redding, J. R. Cannon, J. M. Ryder, and J. A. Purcell, "Cognitive task analysis of expertise in air traffic control," *The International Journal of Aviation Psychology*, vol. 3, no. 4, pp. 257–283, 1993. [Online]. Available: https://doi.org/10.1207/s15327108ijap0304_2

[21] D. J. Simons and R. A. Rensink, "Change blindness: Past, present, and future," *Trends in Cognitive Sciences*, vol. 9, no. 1, pp. 16–20, 2005.

[22] I. Rock, C. M. Linnett, P. Grant, and A. Mack, "Perception without attention: Results of a new method," *Cognitive Psychology*, vol. 24, no. 4, pp. 502–534, 1992.

[23] E. C. M. C. E. Horvitz, "Notification, disruption, and memory: Effects of messaging interruptions on memory and performance," in *Proceedings of Interact*, 2001, p. 263.

[24] N. A. Stanton, R. Stewart, D. Harris, R. J. Houghton, C. Baber, R. McMaster, P. Salmon, G. Hoyle, G. Walker, M. S. Young *et al.*, "Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology," *Ergonomics*, vol. 49, no. 12-13, pp. 1288–1311, 2006.

[25] E. M. Rantanen and B. R. Levinthal, "Effect of air traffic controller taskload and temporal awareness on task prioritization," in *2005 International Symposium on Aviation Psychology*, 2005, p. 601.

[26] W. Stallings, *Operating systems: internals and design principles*. Boston: Prentice Hall,, 2012.

[27] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

BIOGRAPHY

Max Nicosia *received his B.Sc. in Computer Science from the University of St Andrews and his M.Phil in Advanced Computer Science from the University of Cambridge. He is currently a Ph.D. student in the Intelligent Interactive Systems group in the Engineering Design Centre, Department of Engineering at the University of Cambridge. His research interests include attention and context aware-systems and multi-sensing.*

Per Ola Kristensson *is Professor of Interactive Systems Engineering in the Department of Engineering at the University of Cambridge and a Fellow of Trinity College, Cambridge. He leads the Intelligent Interactive Systems group, which belongs to the Engineering Design Centre.*